Pembuatan Program Vigenere Cipher, Playfair Cipher, Affine Cipher, Hill Cipher, Enigma Cipher

Laporan Tugas Kecil 1

Diajukan Untuk Memenuhi Tugas Kecil 1 IF4020 Kriptografi

Semester I 2021/2022

Disusun oleh

Gde Anantha Priharsena **(13519026)**

Reyhan Emyr Arrosyid **(13519167)**

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

BAB I

Implementasi Program

Bahasa Pemrograman : Python

Framework : Flask

1. app.py

```python
from flask import Flask, render_template, request, send_file
from playfair_cipher import getKeywordMatrices, textToBigramArray, encipherBigram, decipherBigram,
encipherBigramToText, decipherBigramToText
from affine_cipher import encryptStringAffine, decryptStringAffine
from vigenere_cipher import vigenere, autoKeyVigenere, fullVigenere, extendedVigenere
from hill_cipher import textToArray, encryptHill, decryptHill, formKeyMatricesFromInput
import io
import re

app = Flask(__name__)

def display5LetterGroup(text):
    result = re.findall('.{1,5}', text)
    result = ' '.join(result)
    return result

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/vigenere/standard/encrypt", methods=['POST','GET'])
def standardVigenereEncrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        cipher_text = vigenere(text, keyword)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("vigenere.html", mode="Encrypt", keyword=keyword, plaintext=text,
result=cipher_text, display=display)
    else:
        return render_template("vigenere.html", mode="Encrypt", display="option1")

@app.route("/vigenere/standard/decrypt", methods=['POST','GET'])
def standardVigenereDecrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        plain_text = vigenere(text, keyword, type="DEC")
        display = request.form['inlineRadio']
        if(display=="option2"):
            plain_text = display5LetterGroup(plain_text)
        return render_template("vigenere.html", mode="Decrypt", keyword=keyword, ciphertext=text,
result=plain_text, display=display)
    else:
        return render_template("vigenere.html", mode="Decrypt", display="option1")
```

```python
@app.route("/vigenere/full/encrypt", methods=['POST','GET'])
def fullVigenereEncrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        cipher_text = fullVigenere(text, keyword)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("full_vigenere.html", mode="Encrypt", keyword=keyword, plaintext=text,
result=cipher_text, display=display)
    else:
        return render_template("full_vigenere.html", mode="Encrypt", display="option1")

@app.route("/vigenere/full/decrypt", methods=['POST','GET'])
def fullVigenereDecrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        plain_text = fullVigenere(text, keyword, type="DEC")
        display = request.form['inlineRadio']
        if(display=="option2"):
            plain_text = display5LetterGroup(plain_text)
        return render_template("full_vigenere.html", mode="Decrypt", keyword=keyword, ciphertext=text,
result=plain_text, display=display)
    else:
        return render_template("full_vigenere.html", mode="Decrypt", display="option1")

@app.route("/vigenere/autokey/encrypt", methods=['POST','GET'])
def autokeyVigenereEncrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        cipher_text = autoKeyVigenere(text, keyword)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("autokey_vigenere.html", mode="Encrypt", keyword=keyword,
plaintext=text, result=cipher_text, display=display)
    else:
        return render_template("autokey_vigenere.html", mode="Encrypt", display="option1")

@app.route("/vigenere/autokey/decrypt", methods=['POST','GET'])
def autokeyVigenereDecrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        plain_text = autoKeyVigenere(text, keyword, type="DEC")
        display = request.form['inlineRadio']
        if(display=="option2"):
            plain_text = display5LetterGroup(plain_text)
        return render_template("autokey_vigenere.html", mode="Decrypt", keyword=keyword,
ciphertext=text, result=plain_text, display=display)
    else:
        return render_template("autokey_vigenere.html", mode="Decrypt", display="option1")

@app.route("/vigenere/extended/encrypt", methods=['POST','GET'])
def extendedVigenereEncrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        file = request.files['file']
        file_contents = file.read()
        filename = file.filename

        cipher_file = extendedVigenere(file_contents, keyword)

        return send_file(cipher_file, as_attachment=True, attachment_filename="encrypted-"+filename)
    else:
        return render_template("extended_vigenere.html", mode="Encrypt")

@app.route("/vigenere/extended/decrypt", methods=['POST','GET'])
def extendedVigenereDecrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        file = request.files['file']
        file_contents = file.read()
        filename = file.filename

        cipher_file = extendedVigenere(file_contents, keyword, type="DEC")

        return send_file(cipher_file, as_attachment=True, attachment_filename="decrypted-"+filename)
    else:
        return render_template("extended_vigenere.html", mode="Decrypt")
```

```python
@app.route("/playfair/encrypt", methods=['POST','GET'])
def playfairEncrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']

        matrix = getKeywordMatrices(keyword)
        cipher_array = encipherBigram(matrix, textToBigramArray(text))
        cipher_text = encipherBigramToText(cipher_array)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("playfair.html", mode="Encrypt", keyword=keyword, plaintext=text,
result=cipher_text, display=display)
    else:
        return render_template("playfair.html", mode="Encrypt", display="option1")

@app.route("/playfair/decrypt", methods=['POST','GET'])
def playfairDecrypt():
    if request.method == 'POST':
        keyword = request.form['text1']
        text = request.form['text2']
        matrix = getKeywordMatrices(keyword)
        decipher_array = decipherBigram(matrix, textToBigramArray(text))
        decipher_text = decipherBigramToText(decipher_array)
        display = request.form['inlineRadio']
        if(display=="option2"):
            decipher_text = display5LetterGroup(decipher_text)
        return render_template("playfair.html", mode="Decrypt", keyword=keyword, ciphertext=text,
result=decipher_text, display=display)
    else:
        return render_template("playfair.html", mode="Decrypt", display="option1")

@app.route("/affine/encrypt", methods=['POST','GET'])
def affineEncrypt():
    offset = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25]
    coprime = [1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25]
    if request.method == 'POST':
        text = request.form['text2']
        m = int(request.form['select1'])
        b = int(request.form['select2'])
        cipher_text = encryptStringAffine(text, m, b)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("affine.html", mode="Encrypt", offset=offset, coprime=coprime,
plaintext=text, result=cipher_text, m=m, b=b, display=display)
    else:
        return render_template("affine.html", mode="Encrypt", offset=offset, coprime=coprime,
display="option1")

@app.route("/affine/decrypt", methods=['POST','GET'])
def affineDecrypt():
    offset = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25]
    coprime = [1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25]
    if request.method == 'POST':
        text = request.form['text2']
        m = int(request.form['select1'])
        b = int(request.form['select2'])
        decipher_text = decryptStringAffine(text, m, b)
        display = request.form['inlineRadio']
        if(display=="option2"):
            decipher_text = display5LetterGroup(decipher_text)
        return render_template("affine.html", mode="Decrypt", offset=offset, coprime=coprime,
ciphertext=text, result=decipher_text, m=m, b=b, display=display)
    else:
        return render_template("affine.html", mode="Decrypt", offset=offset, coprime=coprime,
display="option1")
```

```python
@app.route("/hill/encrypt", methods=['POST','GET'])
def hillEncrypt():
    if request.method == 'POST':
        n = int(request.form['text1'])
        array = request.form['text2']
        entries = list(map(int, array.split()))
        text = request.form['text3']
        matrices= formKeyMatricesFromInput(entries, n)
        cipher_text = encryptHill(matrices, textToArray(text, n), n)
        display = request.form['inlineRadio']
        if(display=="option2"):
            cipher_text = display5LetterGroup(cipher_text)
        return render_template("hill.html", mode="Encrypt", plaintext=text, result=cipher_text, n=n,
array=array, display=display)
    else:
        return render_template("hill.html", mode="Encrypt", display="option1")

@app.route("/hill/decrypt", methods=['POST','GET'])
def hillDecrypt():
    if request.method == 'POST':
        n = int(request.form['text1'])
        array = request.form['text2']
        entries = list(map(int, array.split()))
        text = request.form['text3']
        matrices= formKeyMatricesFromInput(entries, n)
        decipher_text = decryptHill(matrices, textToArray(text, n), n)
        display = request.form['inlineRadio']
        if(display=="option2"):
            decipher_text = display5LetterGroup(decipher_text)
        return render_template("hill.html", mode="Decrypt", ciphertext=text, result=decipher_text, n=n,
array=array, display=display)
    else:
        return render_template("hill.html", mode="Decrypt", display="option1")

@app.route("/saveresult", methods=['POST'])
def saveResult():
    result = request.form['result']

    return send_file(io.BytesIO(result.encode()), mimetype="text/plain",as_attachment=True,
attachment_filename="result.txt")

if __name__ == "__main__":
    app.config['TEMPLATES_AUTO_RELOAD'] = True
    app.run(debug=True,threaded=True)
```

2. vigenere_cipher.py

```python
import io

fullVigenereTable = [
  "NSBEAPLMOHFRVKUIQCJXYZDWGT",
  "VEYKPXHRWODFSUGNAIJTZBMCLQ",
  "MGAXQFVWSZYRLPNEUDBCHITKOJ",
  "QUSYFRCMTXGPJBZOWKLNEHAVDI",
  "EQPVFUXJWBTNOHRAMSIGYCZKDL",
  "GPEVZDXOFTSQAUJBRCWNKILMYH",
  "SPNFXMZBCDIVOYLUJTWHRQAGKE",
  "DVILYGTMUXPQKJAHCBWZONESFR",
  "SDHCXPNOJITAGRLMWZUBKFYVQE",
  "UQFJWDVOBLNYCERMXKSPHZAIGT",
  "YKNHQUSWRCODILJZEAFBMXTGPV",
  "XWSBRTJFYEKZQHPNLDCGMAUVIO",
  "MJRGHKTECLBSXVZOQYDNWPIAFU",
  "RIQWONPJEMGLBUCDKSFYXHAZVT",
  "JWAGCKLVMBZOXPQEUSNYTHIDFR",
  "BKJPEWXFHICMLYANVRUTZSQOGD",
  "CQJMHASEUFWNILKXTYVGBZORPD",
  "MEPYKASWJVGQHUOIBTCRFZDNXL",
  "DUKLRQMECHPGFOIZABYVNXSJWT",
  "AMVYWLSIXKUNDFJORPQTBHGCEZ",
  "OZJPHBQNUMCIYKLTDERFWAXVSG",
  "XGUBYKTLWSMHOVINACJZFDPRQE",
  "CFMYVSEXRGUNWIBOLKHQTPDZJA",
  "NTMWBEDIOHJSKLXFUQRVGCPYAZ",
  "ITXOAHFVSDLWZMPQCNBJYRKGUE",
  "YFUWSAJQHZTBEODRCXMGIVNPKL",
]
```

```python
# Convert alfabet ke angka (case insensitive, a=A=0...)
def alphabetToInt(char):
  if char.isupper():
    return ord(char) - 65
  else:
    return ord(char) - 97

# Fungsi standard vigenere cypher
def vigenere(inText, key, type="ENC"):
  text = ''.join(filter(str.isalpha, inText))
  key = ''.join(filter(str.isalpha, key))
  if type == "ENC":
    outText = ""
    for i in range(len(text)):
      outText += chr((alphabetToInt(text[i]) + alphabetToInt(key[i % len(key)])) % 26 + 65)
    return outText
  elif type == "DEC":
    outText = ""
    for i in range(len(text)):
      outText += chr((alphabetToInt(text[i]) - alphabetToInt(key[i % len(key)])) % 26 + 65)
    return outText
  else:
    return "Type not valid"

# Fungsi full vigenere cypher
def fullVigenere(inText, key, type="ENC"):
  text = ''.join(filter(str.isalpha, inText))
  key = ''.join(filter(str.isalpha, key))

  if type == "ENC":
    outText = ""
    for i in range(len(text)):
      outText += fullVigenereTable[alphabetToInt(key[i % len(key)])][alphabetToInt(text[i])]
    return outText
  elif type == "DEC":
    outText = ""
    for i in range(len(text)):
      outText += chr(fullVigenereTable[alphabetToInt(key[i % len(key)])].index(text[i]) + 65)
    return outText
  else:
    return "Type not valid"

# Fungsi auto key vigenere cypher
def autoKeyVigenere(inText, key, type="ENC"):
  text = ''.join(filter(str.isalpha, inText))
  key = ''.join(filter(str.isalpha, key))

  if type == "ENC":
    outText = ""
    key += text
    for i in range(len(text)):
      outText += chr((alphabetToInt(text[i]) + alphabetToInt(key[i % len(key)])) % 26 + 65)
    return outText
  elif type == "DEC":
    outText = ""
    for i in range(len(text)):
      outText += chr((alphabetToInt(text[i]) - alphabetToInt(key[i % len(key)])) % 26 + 65)
      key += outText[-1]
    return outText
  else:
    return "Type not valid"

# Fungsi extended vigenere cypher
def extendedVigenere(inText, key, type="ENC"):
  if type == "ENC":
    outText = bytearray()
    for i in range(len(inText)):
      outText.append(((inText[i]) + ord(key[i % len(key)])) % 256)
    return io.BytesIO(outText)
  elif type == "DEC":
    outText = bytearray()
    for i in range(len(inText)):
      outText.append(((inText[i]) - ord(key[i % len(key)])) % 256)
    return io.BytesIO(outText)
  else:
    return "Type not valid"
```

3. playfair_cipher.py

```python
import re
from collections import OrderedDict

def getKeywordMatrices(keyword):
    alphabet =
['A','B','C','D','E','F','G','H','I','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
    regex_keyword = re.sub(r'[^a-zA-Z]', '', keyword).upper()
    regex_keyword_wo_j = re.sub(r'[J]', '', regex_keyword)
    unique_capital_keyword = ''.join(OrderedDict.fromkeys(regex_keyword_wo_j))

    i,j,k=0,0,0

    matrix = [['~' for i in range(5)] for j in range(5)]
    while i < 5:
        while j<5:
            if((i*5)+j+1)<=len(unique_capital_keyword):
                matrix[i][j] = unique_capital_keyword[i*5+j]
                j+=1
            else:
                if(alphabet[k] not in (item for sublist in matrix for item in sublist)):
                    matrix[i][j] = alphabet[k]
                    j+=1
                k+=1
        j=0
        i+=1
    return matrix

def addXToRepeatedChar(text) :
    return 'X'.join(text[i:i+1] for i in range(0, len(text)))

def textToBigramArray(text):
    regex_text = re.sub(r'[^a-zA-Z]', '', text).upper()
    regex_text_wo_j = re.sub(r'[J]', 'I', regex_text)

    array_of_repeated = [m.group(0) for m in re.finditer(r'(.)\1*', regex_text_wo_j)]
    bigram_text = ""
    for x in array_of_repeated:
        if(len(x)>1):
            bigram_text+=addXToRepeatedChar(x)
        else:
            bigram_text+=x
    if(len(bigram_text)%2==1):
        bigram_text+="X"
    bigram_array = [bigram_text[i:i+2] for i in range(0, len(bigram_text), 2)]
    return bigram_array

def searchMatrixIndex(char, matrix):
    for i in range(5):
        for j in range(5):
            if(matrix[i][j]==char):
                return [i, j]

def encipherBigram(matrix, bigramArray):
    cipher_array = []
    for bigram in bigramArray:
        index1 = searchMatrixIndex(bigram[0], matrix)
        index2 = searchMatrixIndex(bigram[1], matrix)
        if(index1[0]==index2[0]):
            res = matrix[index1[0]][(index1[1]+1)%5]+matrix[index2[0]][(index2[1]+1)%5]
            cipher_array.append(res)
        elif(index1[1]==index2[1]):
            res = matrix[(index1[0]+1)%5][index1[1]]+matrix[(index2[0]+1)%5][index2[1]]
            cipher_array.append(res)
        else:
            res = matrix[index1[0]][index2[1]]+matrix[index2[0]][index1[1]]
            cipher_array.append(res)
    return cipher_array

def decipherBigram(matrix, bigramArray):
    decipher_array = []
    for bigram in bigramArray:
        index1 = searchMatrixIndex(bigram[0], matrix)
        index2 = searchMatrixIndex(bigram[1], matrix)
        if(index1[0]==index2[0]):
            res = matrix[index1[0]][(index1[1]-1)%5]+matrix[index2[0]][(index2[1]-1)%5]
            decipher_array.append(res)
        elif(index1[1]==index2[1]):
            res = matrix[(index1[0]-1)%5][index1[1]]+matrix[(index2[0]-1)%5][index2[1]]
            decipher_array.append(res)
        else:
            res = matrix[index1[0]][index2[1]]+matrix[index2[0]][index1[1]]
            decipher_array.append(res)
    return decipher_array

def encipherBigramToText(encipherArray):
    return ''.join(encipherArray)

def decipherBigramToText(decipherArray):
    for i in range (len(decipherArray)):
        if(decipherArray[i].count('X') == 1):
            decipherArray[i] = re.sub(r'[X]', '', decipherArray[i])
        elif(decipherArray[i].count('X') == 2):
            decipherArray[i] = "X"
    return ''.join(decipherArray)
```

4. affine_cipher.py

```
import re
alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
']
prime26 = [1,3,5,7,9,11,15,17,19,21,23,25]

def inversOfXModuloN(x, n):
    a = 1
    while(a%x != 0):
        a+=n
    return (a//x)%n

def encryptCharAffine(m, p, b, n):
    return (m*p+b)%n

def decryptCharAffine(invM, c, b, n):
    return (invM*(c-b))%n

def encryptStringAffine(string, m, b):
    alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
']
    regex_text = re.sub(r'[^a-zA-Z]', '', string).upper()
    result = ""
    for char in regex_text:
        result+= alphabet[encryptCharAffine(m, alphabet.index(char), b, 26)]
    return result

def decryptStringAffine(string, m, b):
    alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
']
    regex_text = re.sub(r'[^a-zA-Z]', '', string).upper()
    result = ""
    for char in regex_text:
        result+= alphabet[decryptCharAffine(inversOfXModuloN(m, 26), alphabet.index(char), b, 26)]
    return result
```

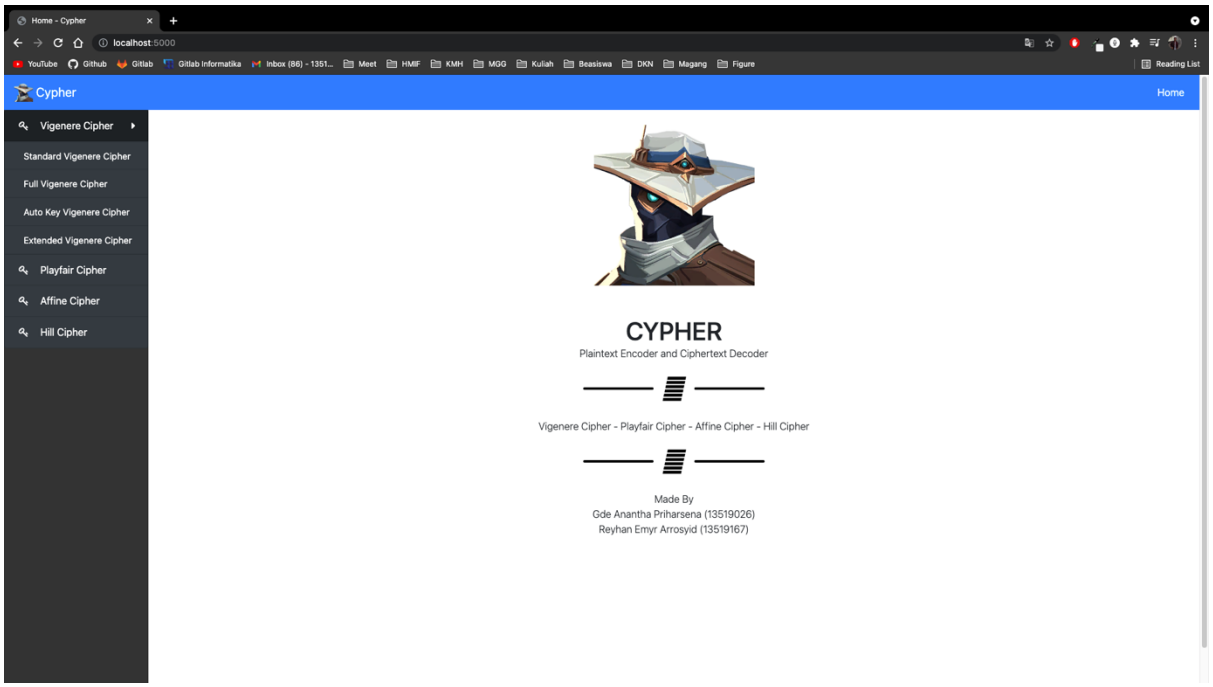
5. hill_cipher.py

```python
import numpy as np
import re
from sympy import Matrix

def inversOfXModuloN(x, n):
    a = 1
    while(a%x != 0):
        a+=n
    return (a//x)%n

def formKeyMatricesFromInput(entry, n):
    matrices= np.array(entry).reshape(n, n)
    return matrices

def textToArray(text, n):
    regex_text = re.sub(r'[^a-zA-Z]', '', text).upper()
    if(len(regex_text)%n != 0):
        regex_text+='X'*(n-(len(regex_text)%n))
    res = [regex_text[i:i+n] for i in range(0, len(regex_text), n)]
    return res

def encryptHill(matrices, arr, n):
    alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
']
    text = ""
    for x in arr:
        current = []
        for i in range(n):
            current.append(alphabet.index(x[i]))
        encryptCurr = np.matmul(matrices, np.array(current)).tolist()
        for i in range(n):
            text+=alphabet[encryptCurr[i]%26]
    return text

def inverseModMatrices(matrices, n):
    pre_inverse1 = matrices.flatten().tolist()
    pre_inverse2 = Matrix(n, n, pre_inverse1)
    pre_inverse3 = pre_inverse2.inv_mod(26)
    inverse = np.array(pre_inverse3)
    return inverse

def decryptHill(matrices, arr, n):
    alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
']
    text = ""
    inverse = inverseModMatrices(matrices, n)
    for x in arr:
        current = []
        for i in range(n):
            current.append(alphabet.index(x[i]))
        encryptCurr = np.matmul(inverse, np.array(current)).tolist()
        for i in range(n):
            text+=alphabet[encryptCurr[i]%26]
    return text
```

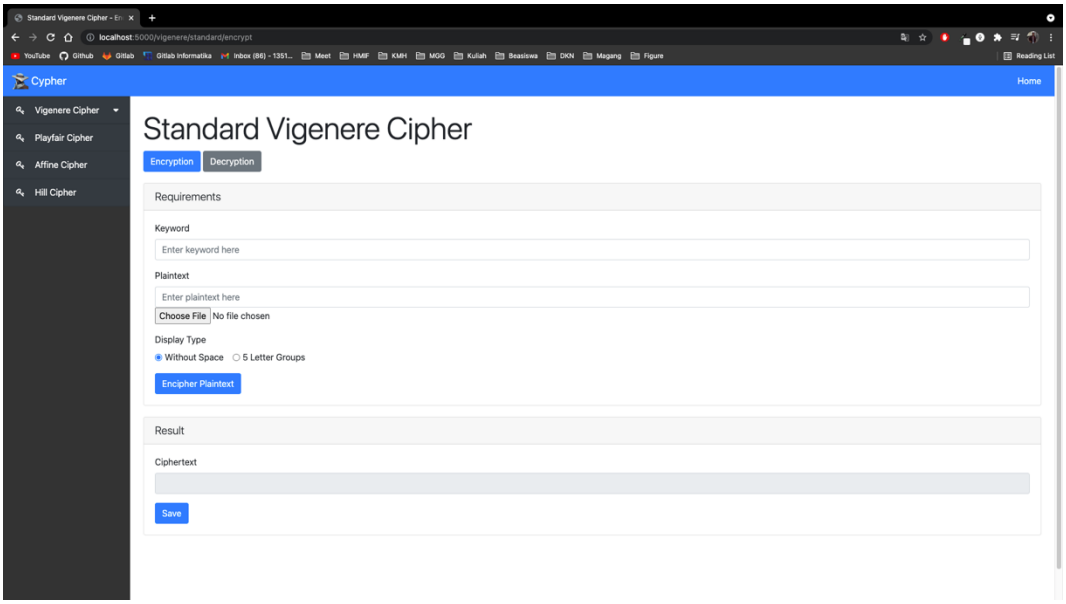
BAB II

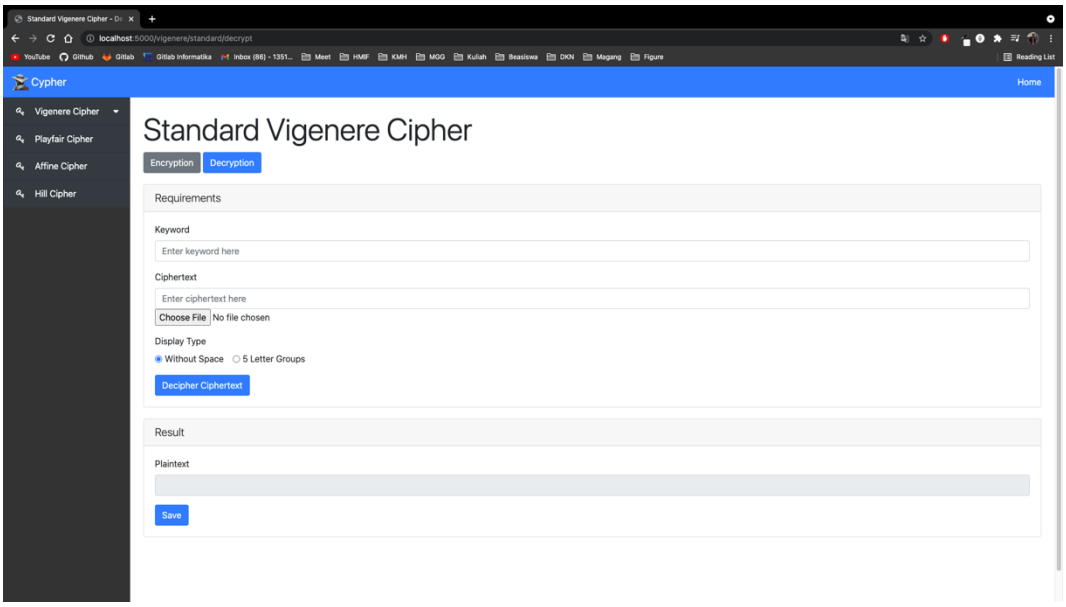
Tampilan Antarmuka Program

1. Home + Sidebar

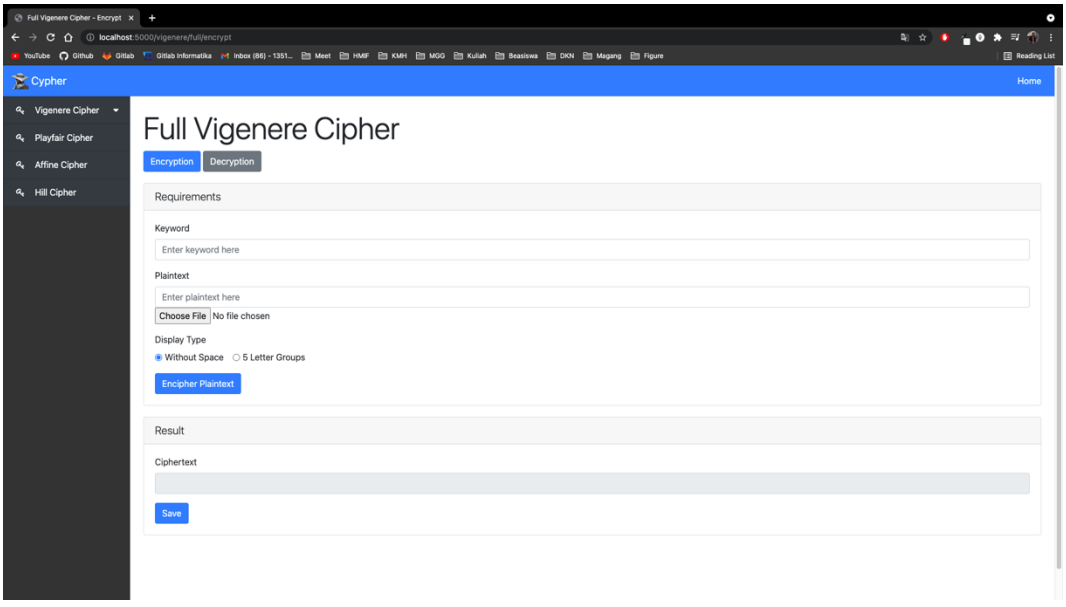


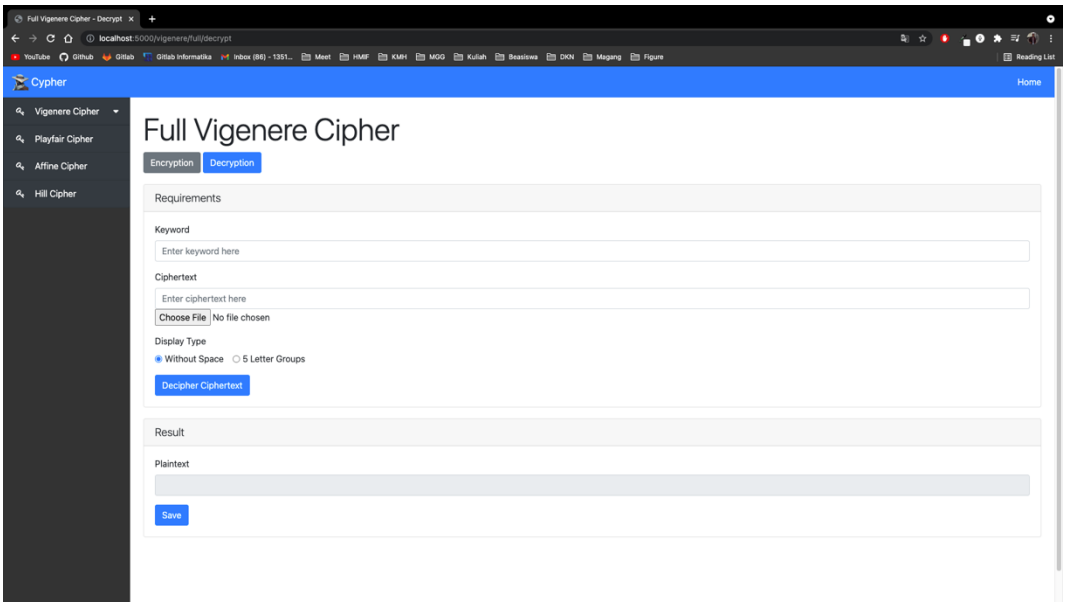
2. Standard Vigenere Cipher

Encryption


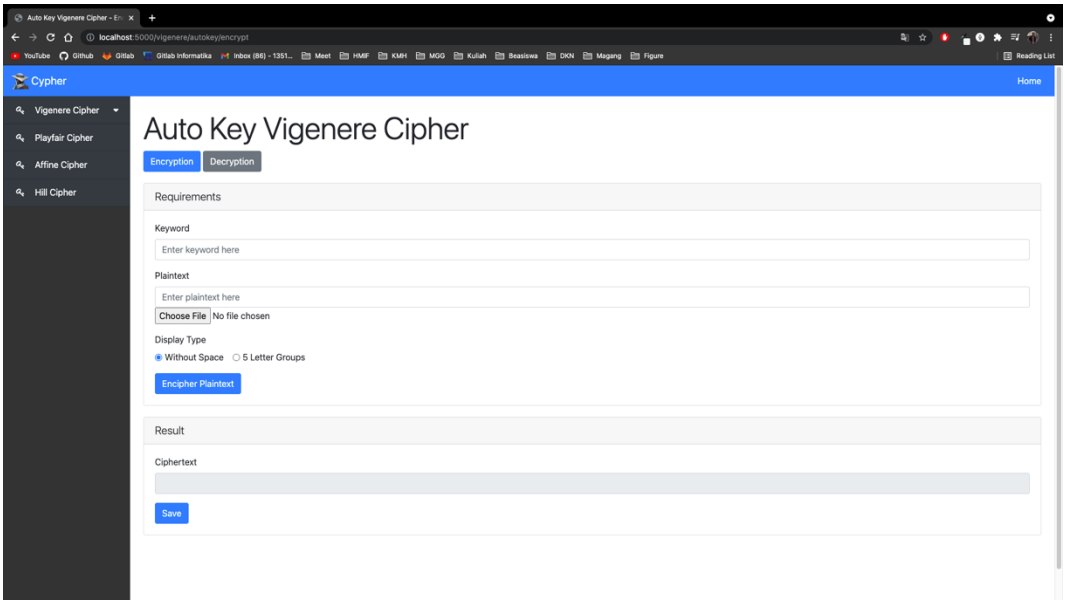
Decryption


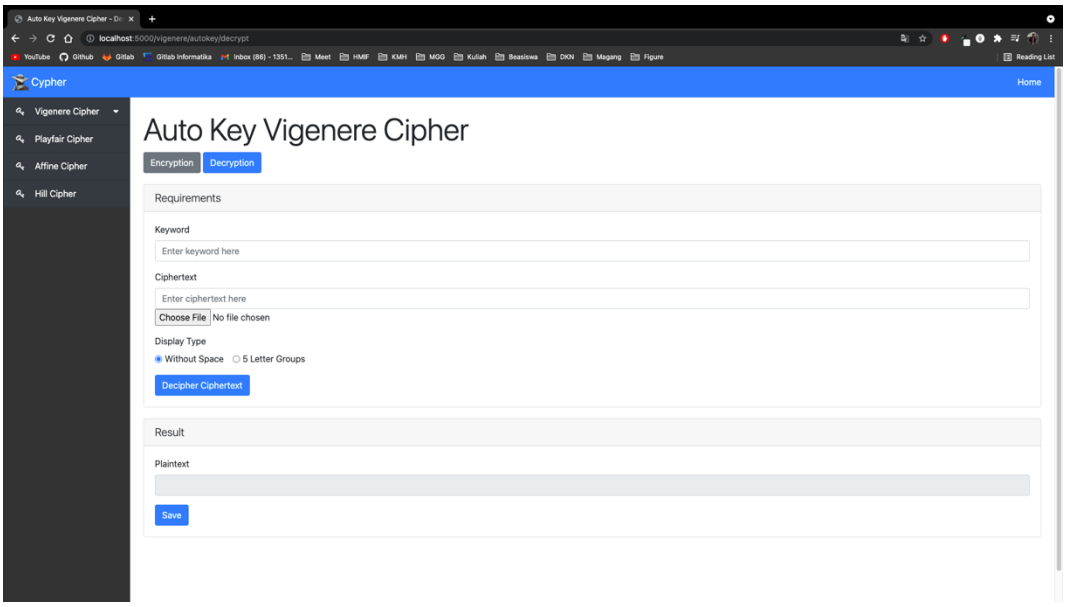
3. Full Vigenere Cipher

Encryption


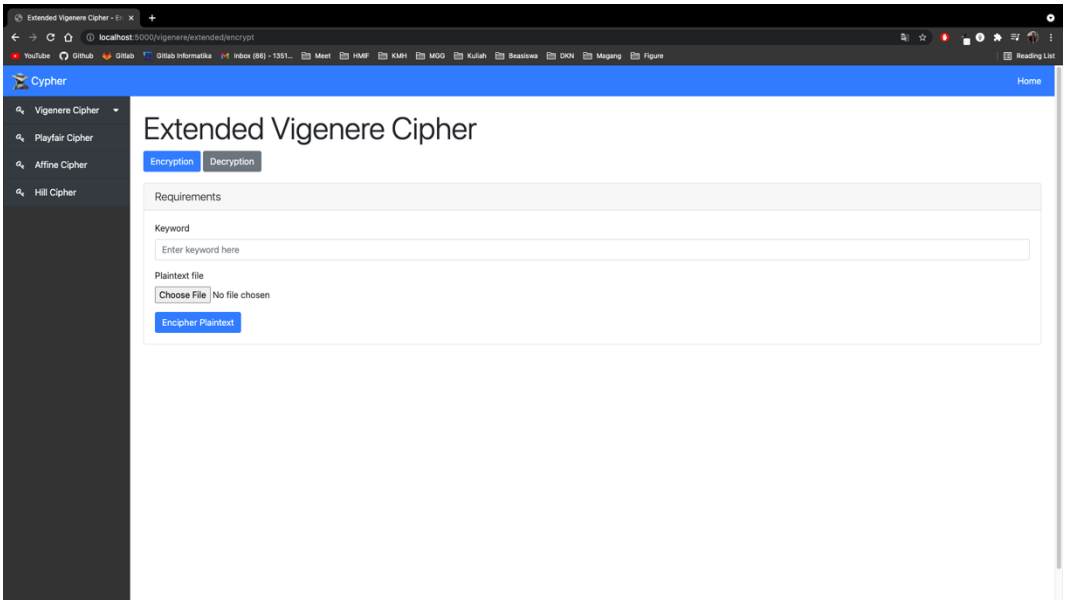
Decryption


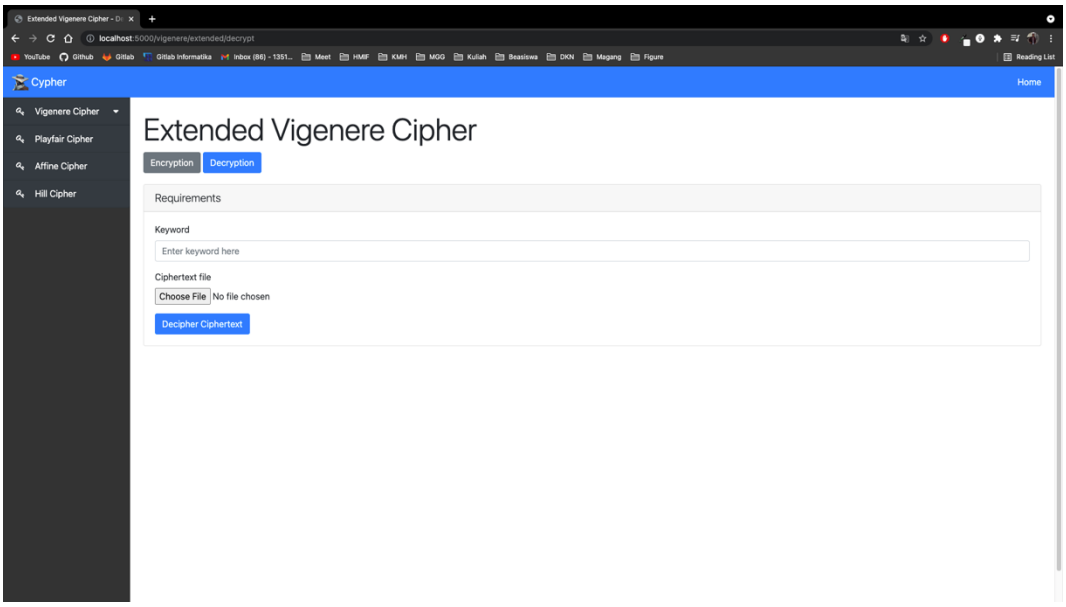
4. Auto Key Vigenere Cipher

Encryption


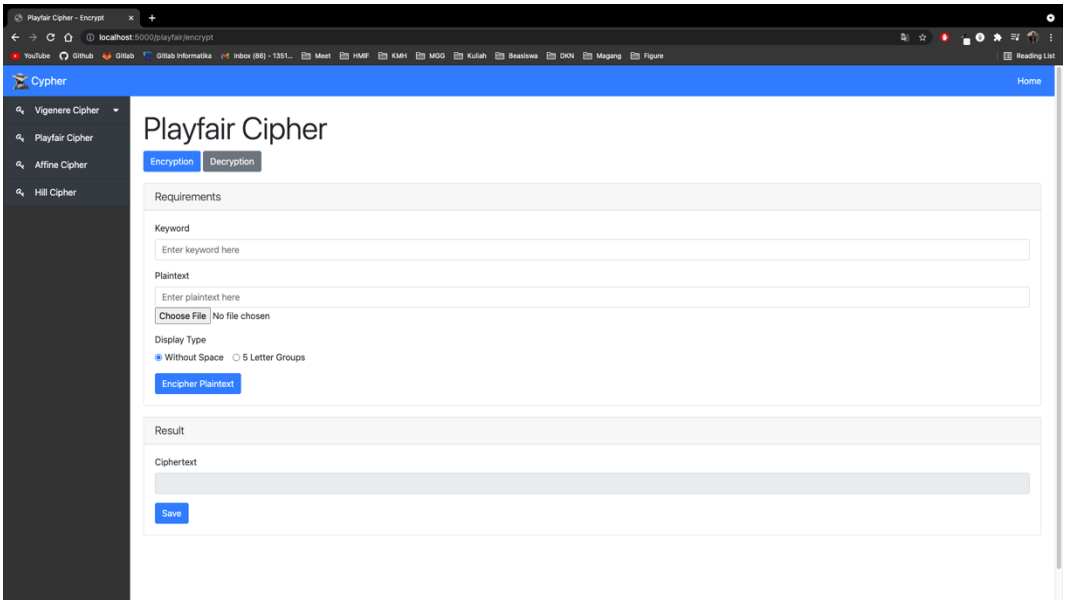
Decryption


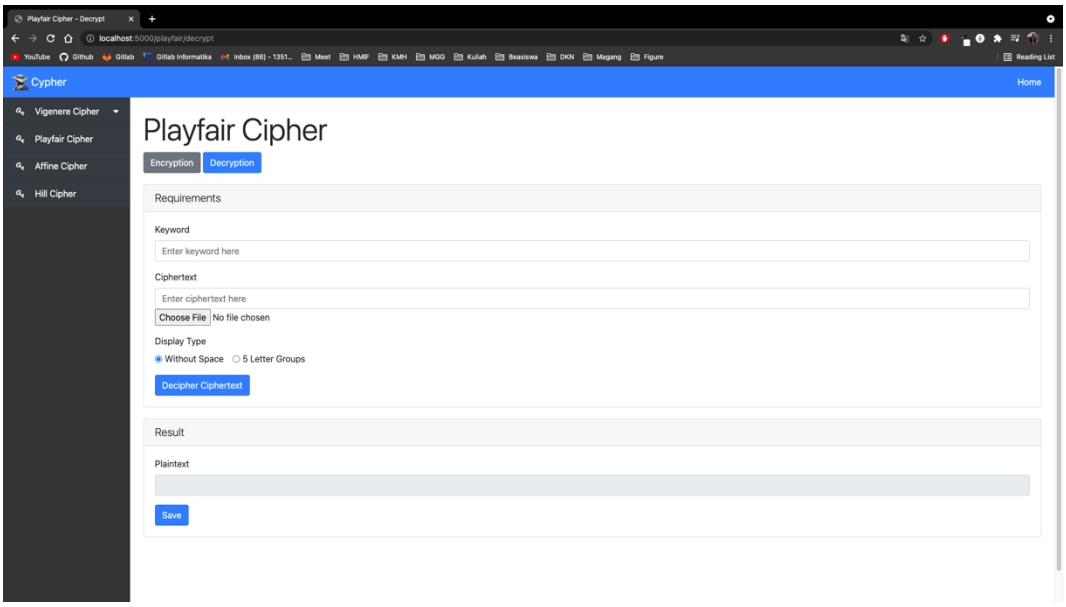
5. Extended Vigenere Cipher

Encryption


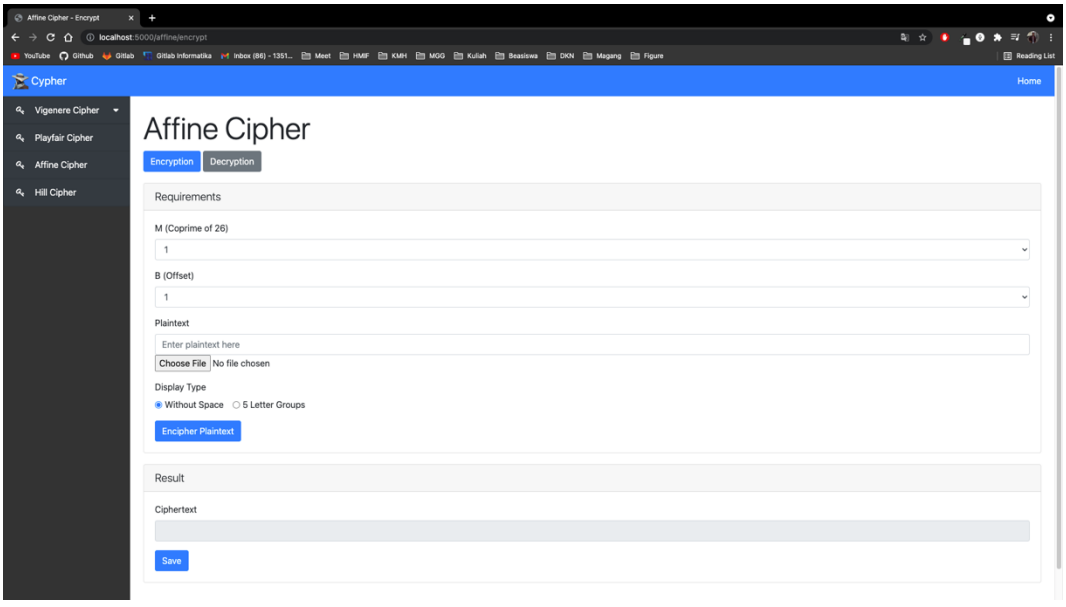
Decryption


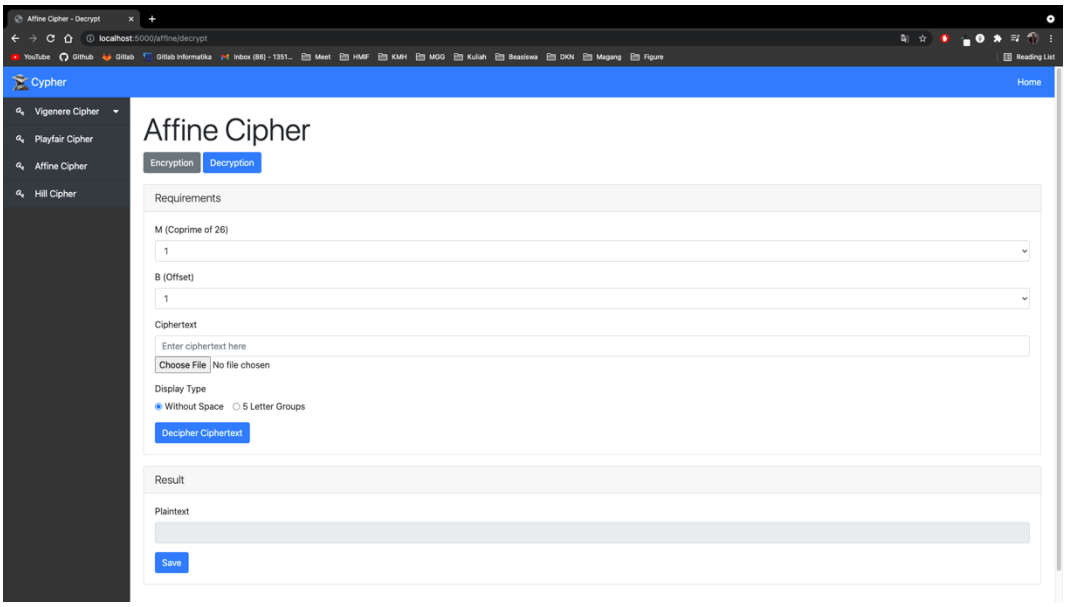
6. Playfair Cipher

Encryption


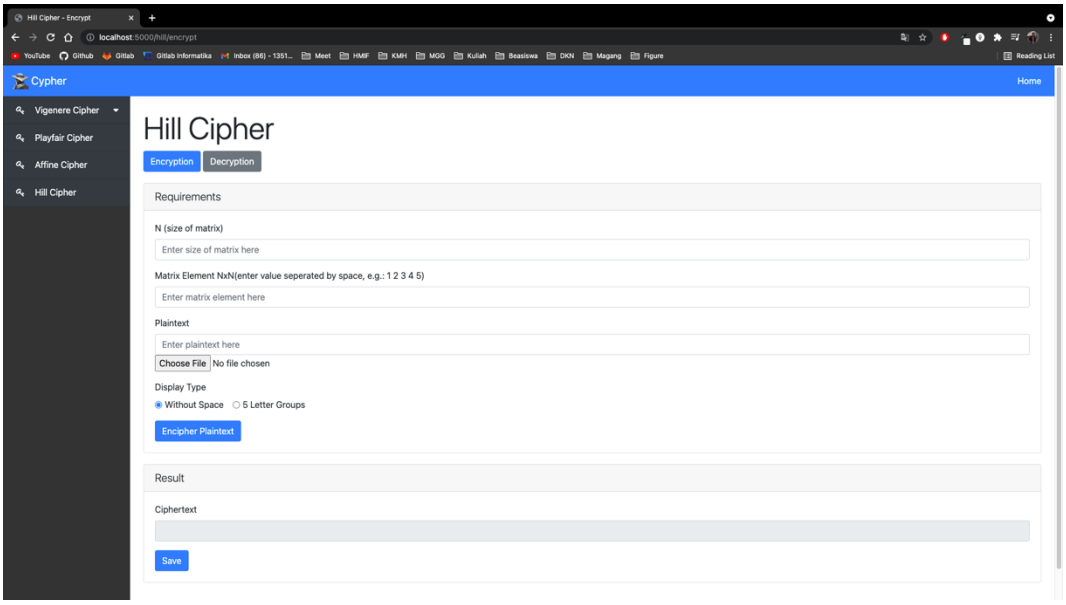
Decryption


7. Affine Cipher

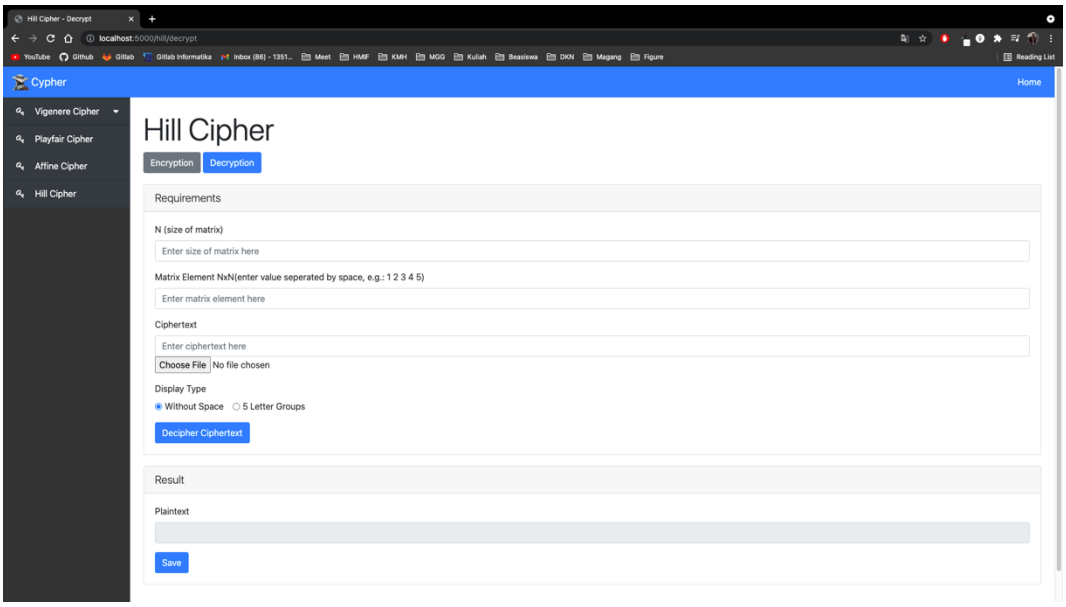
Encryption


Decryption


8. Hill Cipher

Encryption


Decryption


Hill Cipher - Decrypt
localhost:5000/hill/decrypt
Cypher
Home
Vigenere Cipher
Playfair Cipher
Affine Cipher
Hill Cipher
Hill Cipher
Encryption
Decryption
Requirements
N (size of matrix)
Enter size of matrix here
Matrix Element NxN(enter value seperated by space, e.g.: 1 2 3 4 5)
Enter matrix element here
Ciphertext
Enter ciphertext here
Choose File
No file chosen
Display Type
Without Space
5 Letter Groups
Decipher Ciphertext
Result
Plaintext
Save

BAB III

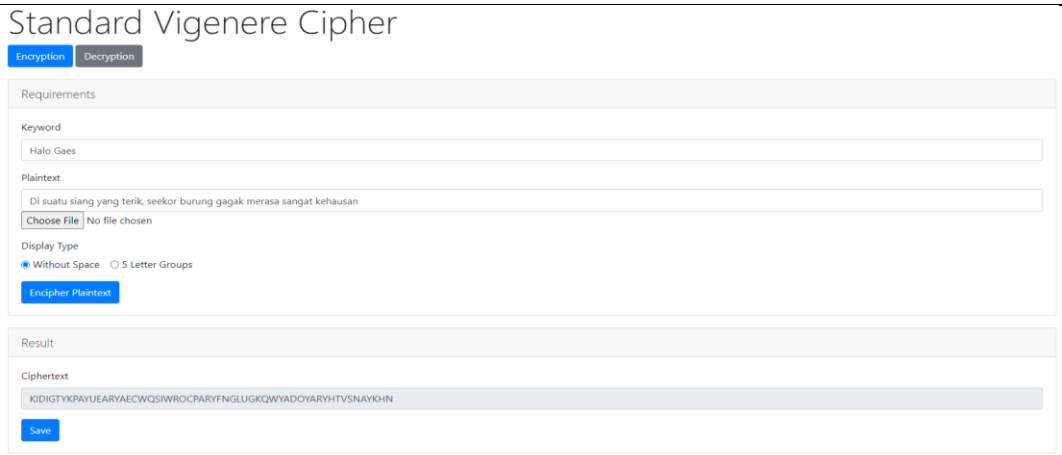
Hasil Percobaan

1. Standard Vigenere Cipher

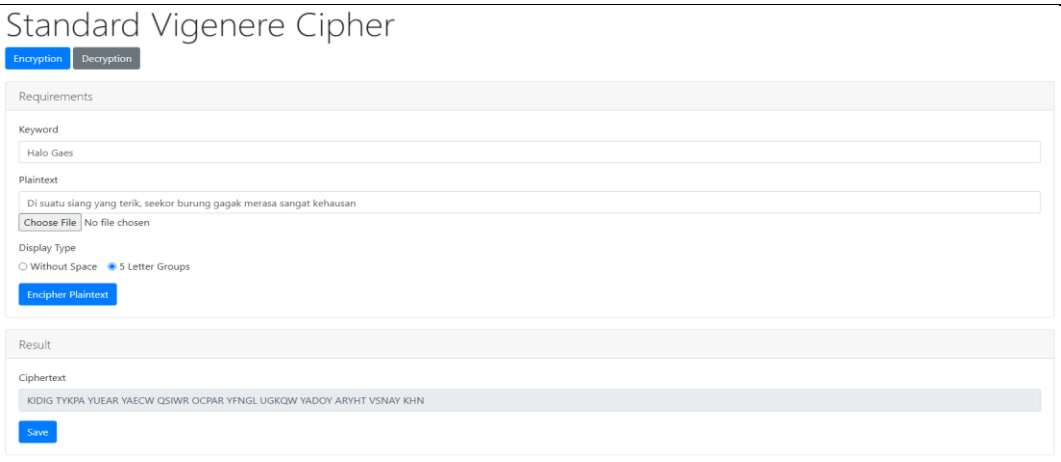
a. Enkripsi

Kata Kunci	Halo Gaes
Plaintext	Di suatu siang yang terik, seekor burung gagak merasa sangat kehausan

Hasil Tanpa Spasi



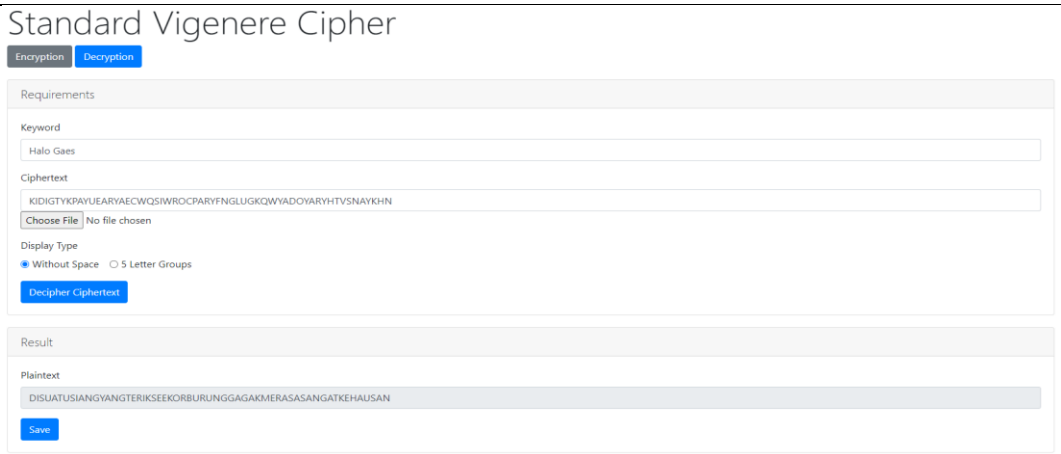
Hasil dalam Kelompok 5 Huruf



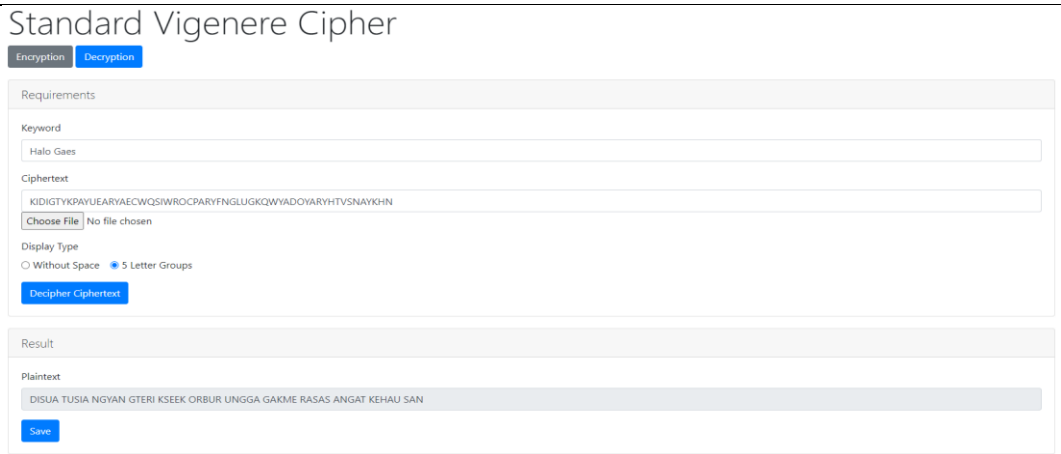
b. Dekripsi

Kata Kunci	Halo Gaes
Ciphertext	KIDIGTYKPAYUEARYAECWQSIWROCPARYFNGLUGKQWYADOYARYHTVSNAYKHN

Hasil Tanpa Spasi



Hasil dalam Kelompok 5 Huruf



2. Full Vigenere Cipher

a. Enkripsi

Kata Kunci	Rusa dan Pemburu
Plaintext	Andi adalah seorang mahasiswa jurusan Teknik Informatika di salah satu Perguruan Tinggi favorit di Bandung

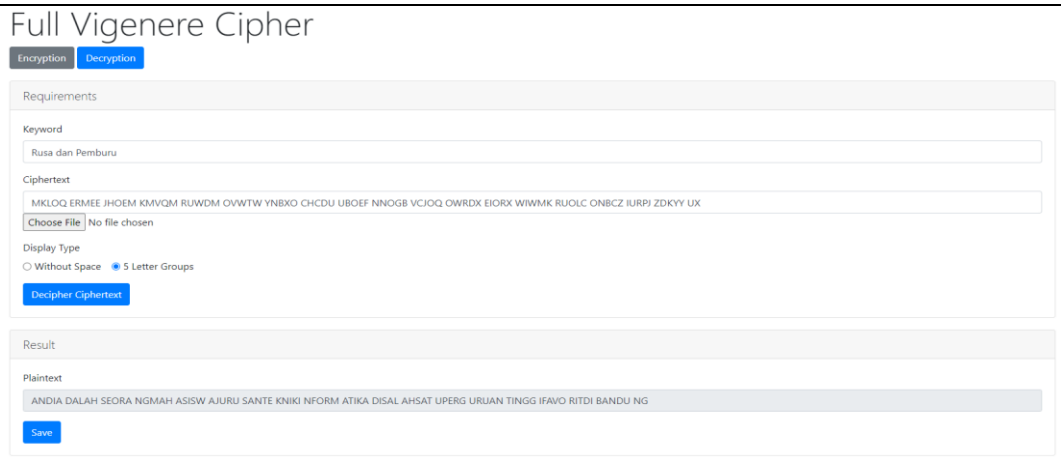
Hasil Tanpa Spasi

Hasil dalam Kelompok 5 Huruf

b. Dekripsi

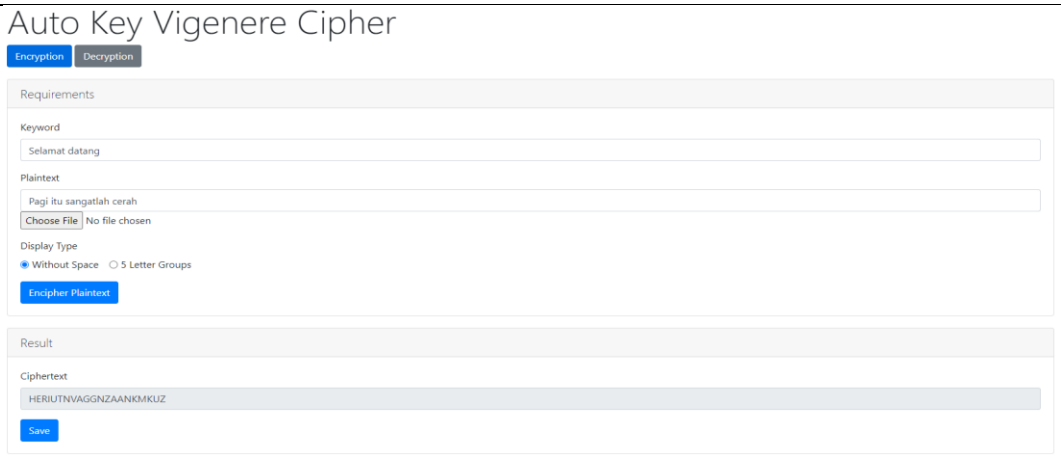
Kata Kunci	Rusa dan Pemburu
Ciphertext	MKLOQ ERMEE JHOEM KMVQM RUWDM OVWTW YNBXO CHCDU UBOEF NNOGB VCJOQ OWRDX EIORX WIWMK RUOLC ONBCZ IURPJ ZDKYY UX

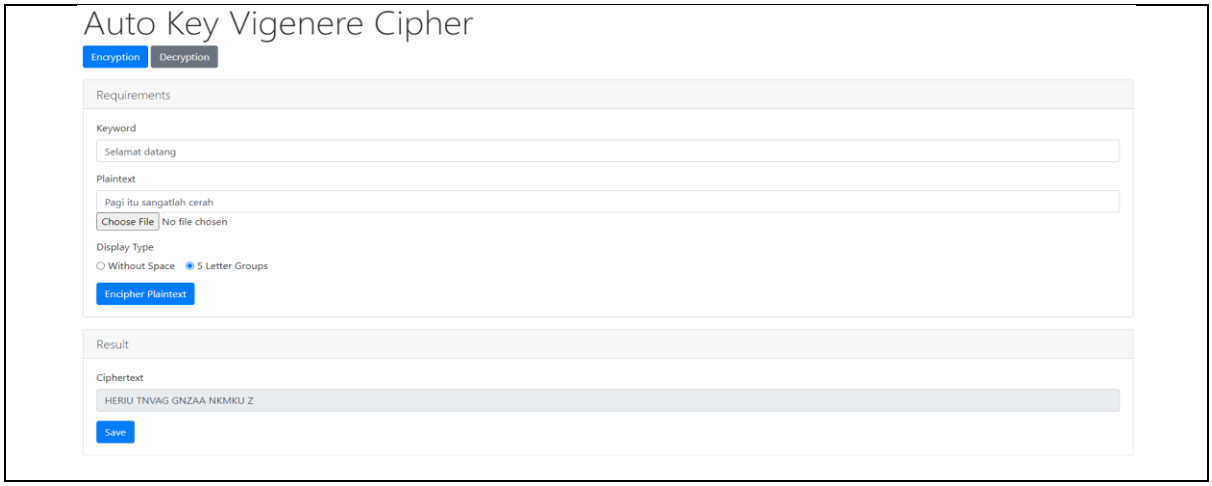
Hasil Tanpa Spasi

Hasil dalam Kelompok 5 Huruf


3. Auto Key Vigenere Cipher

a. Enkripsi

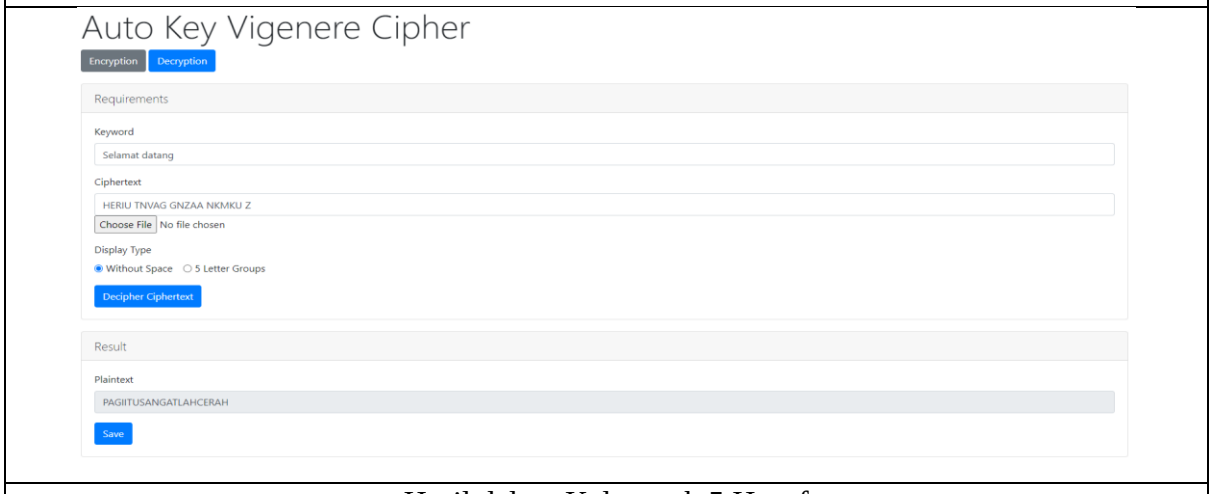
Kata Kunci	Selamat datang
Plaintext	Pagi itu sangatlah cerah
Hasil Tanpa Spasi	
	
Hasil dalam Kelompok 5 Huruf	



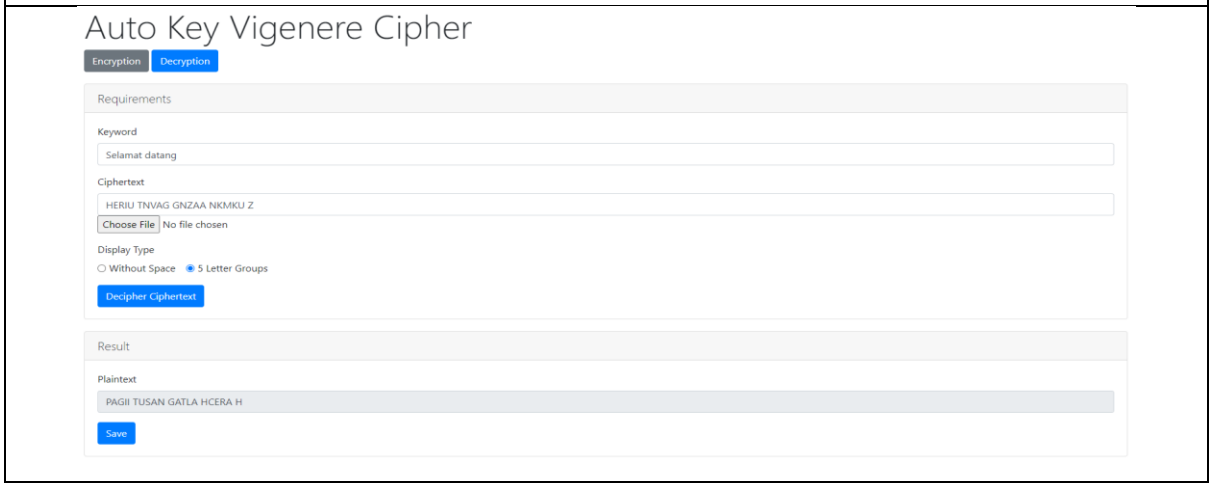
b. Dekripsi

Kata Kunci	Selamat datang
Ciphertext	HERIU TNVAG GNZAA NKMKU Z

Hasil Tanpa Spasi

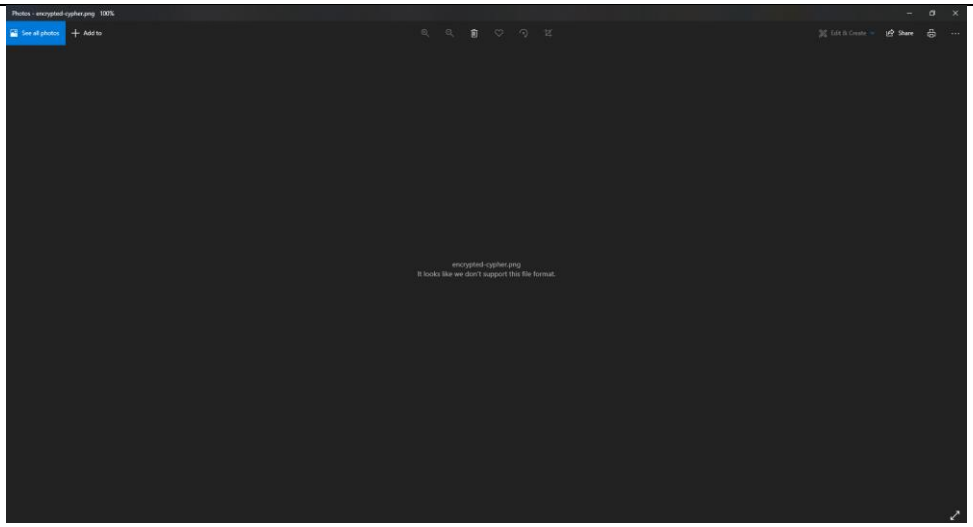


Hasil dalam Kelompok 5 Huruf



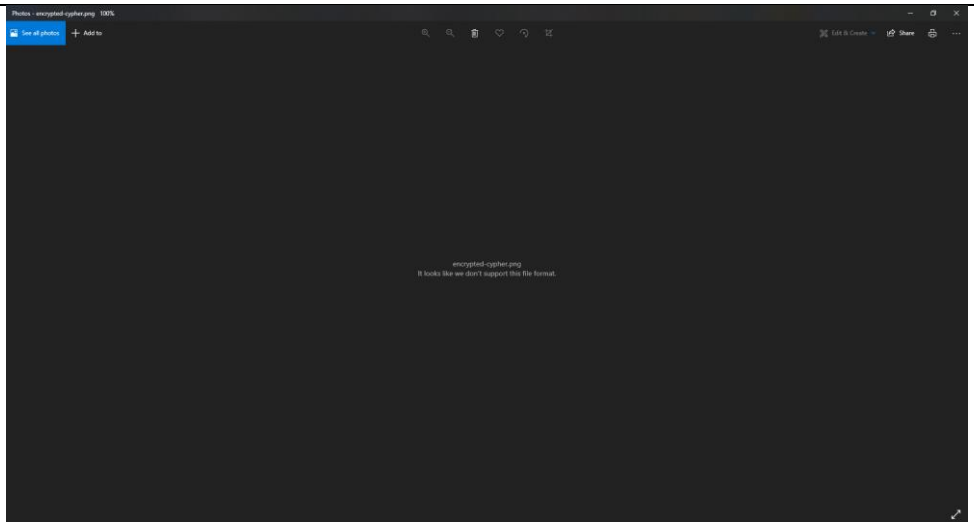
4. Extended Vigenere Cipher

a. Enkripsi

Kata Kunci	Biskuit bakar
Plaintext	 cypher.png
Hasil	
 encrypted-cypher.png	

b. Dekripsi

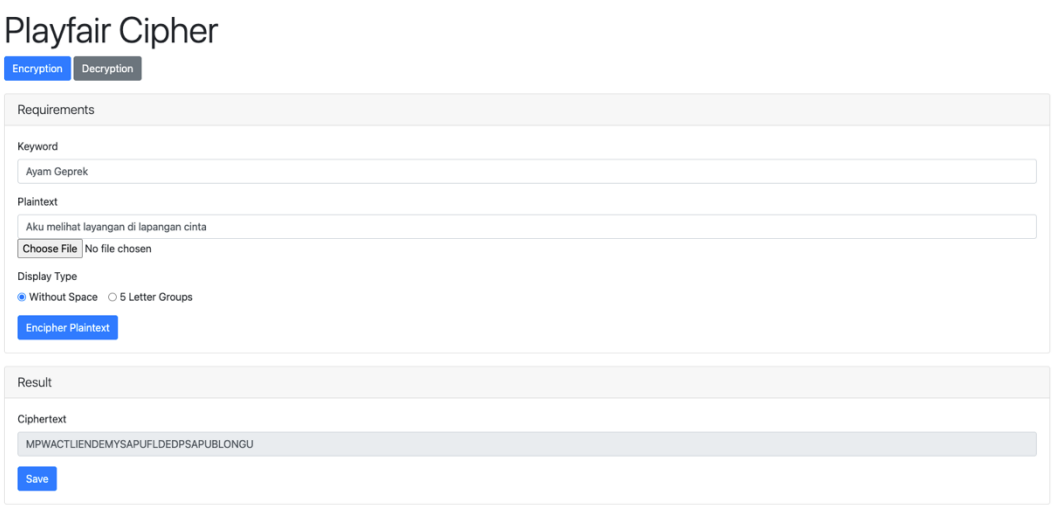
Kata Kunci	Biskuit bakar

Plaintext	 encrypted-cypher.png
Hasil	
 decrypted-encrypted-cypher.png	

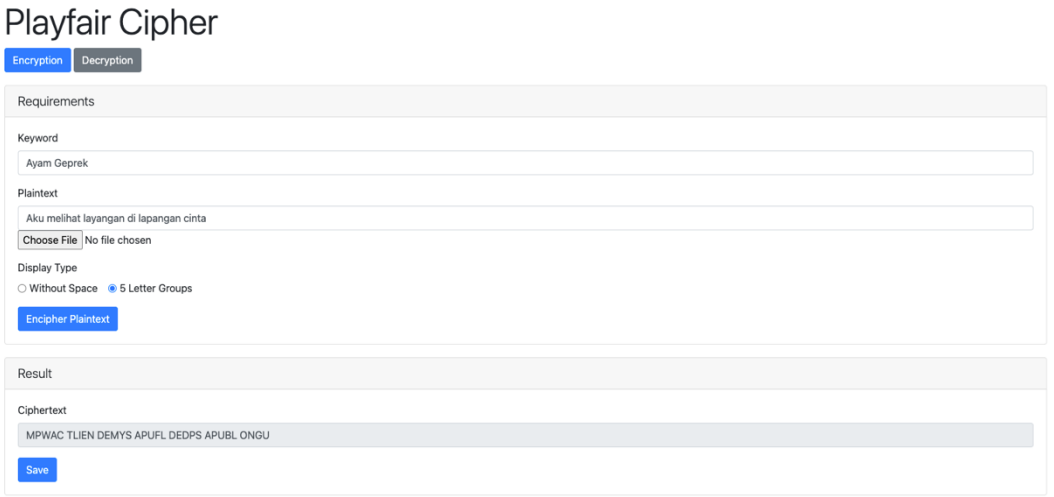
5. Playfair Cipher

a. Enkripsi

Kata Kunci	Ayam Geprek
Plaintext	Aku mau jalan kaki di ITB
Hasil Tanpa Spasi	



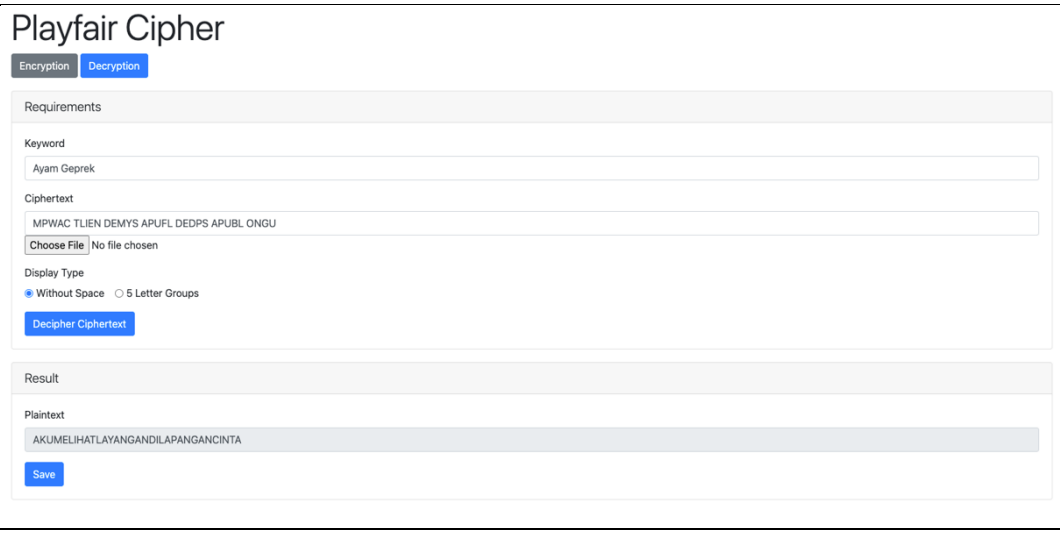
Hasil dalam Kelompok 5 Huruf



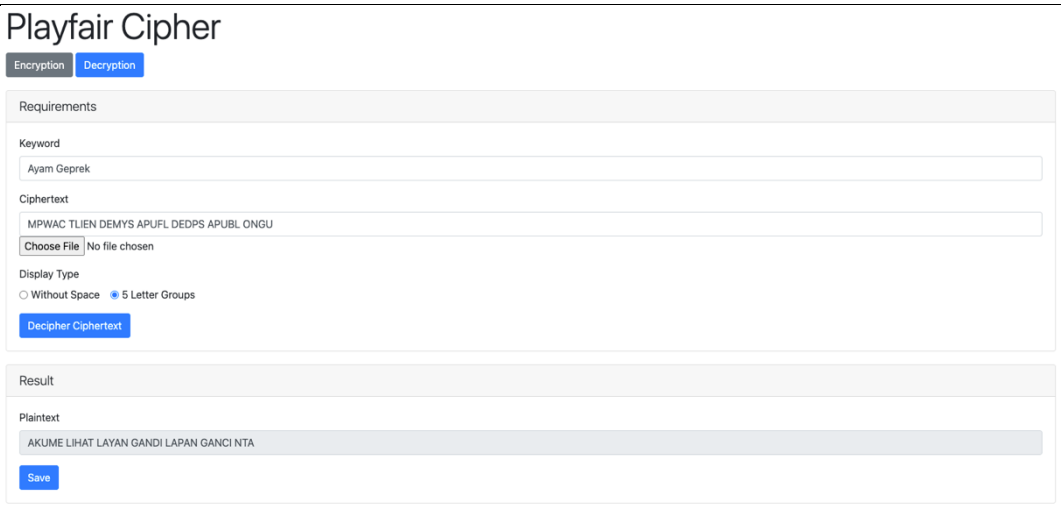
b. Dekripsi

Kata Kunci	Ayam Geprek
Ciphertext	MPWAC TLIEN DEMYS APUFL DEDPS APUBL ONGU

Hasil Tanpa Spasi



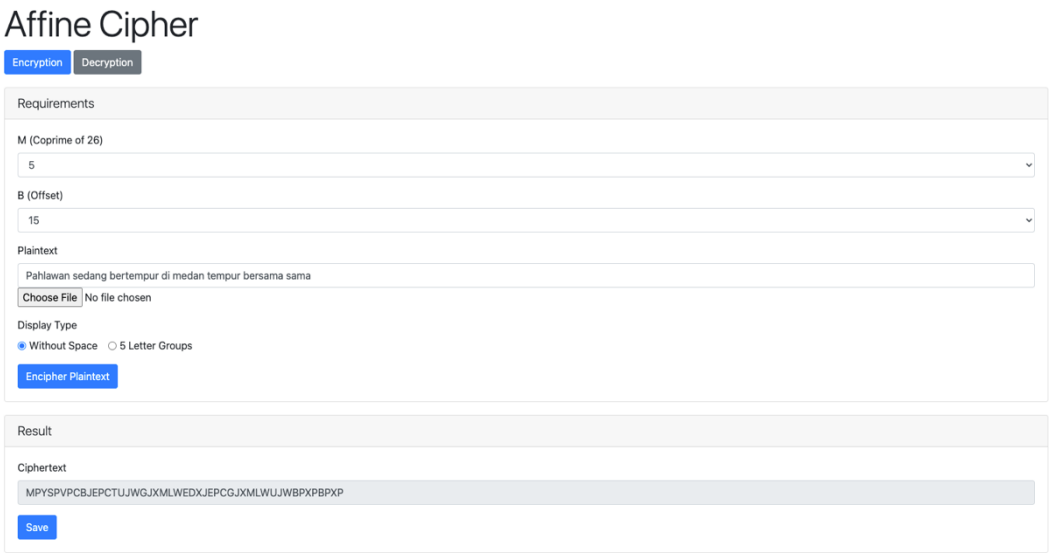
Hasil dalam Kelompok 5 Huruf



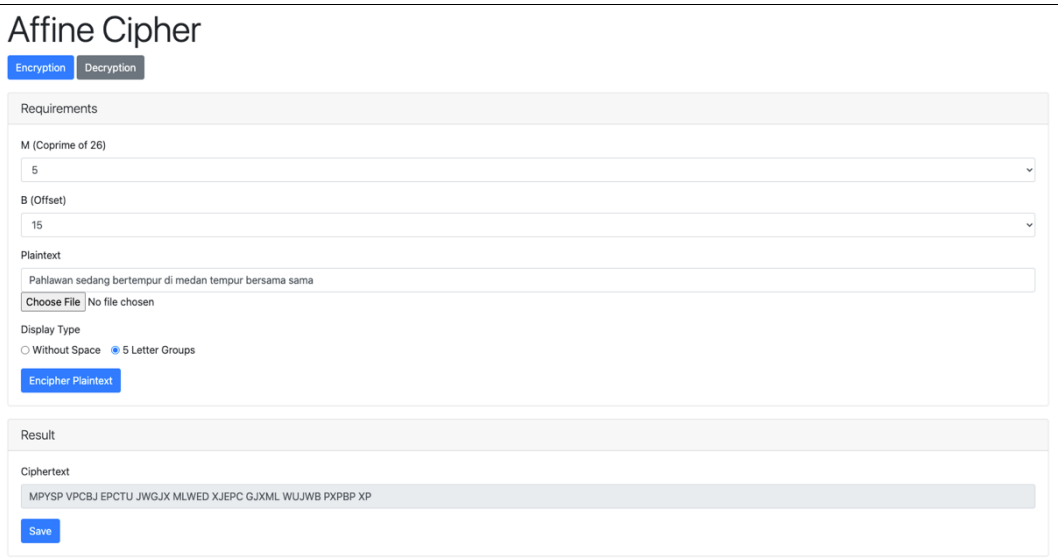
6. Affine Cipher

a. Enkripsi

M	5
B	15
Plaintext	Pahlawan sedang bertempur di medan tempur bersama sama
Hasil Tanpa Spasi	



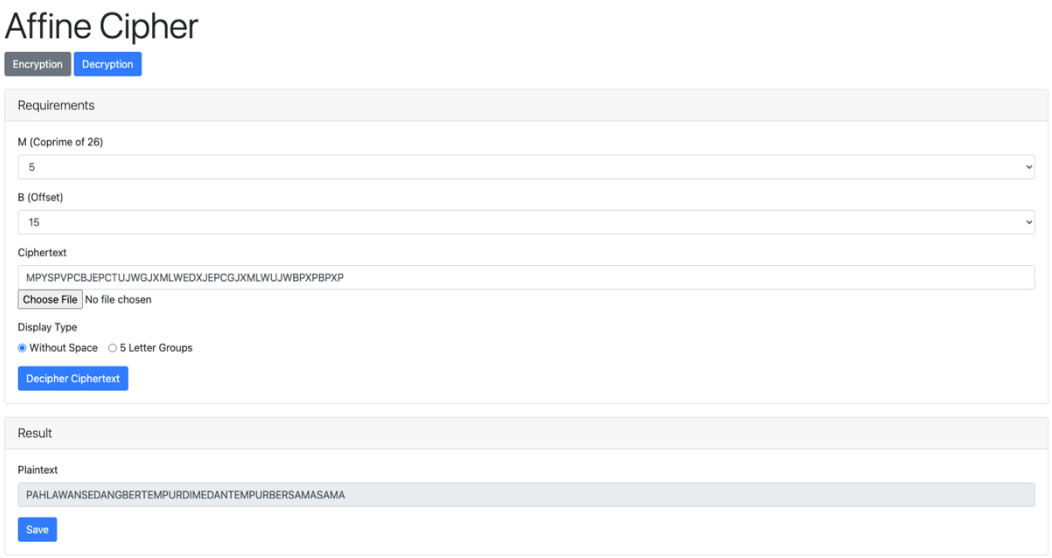
Hasil dalam Kelompok 5 Huruf



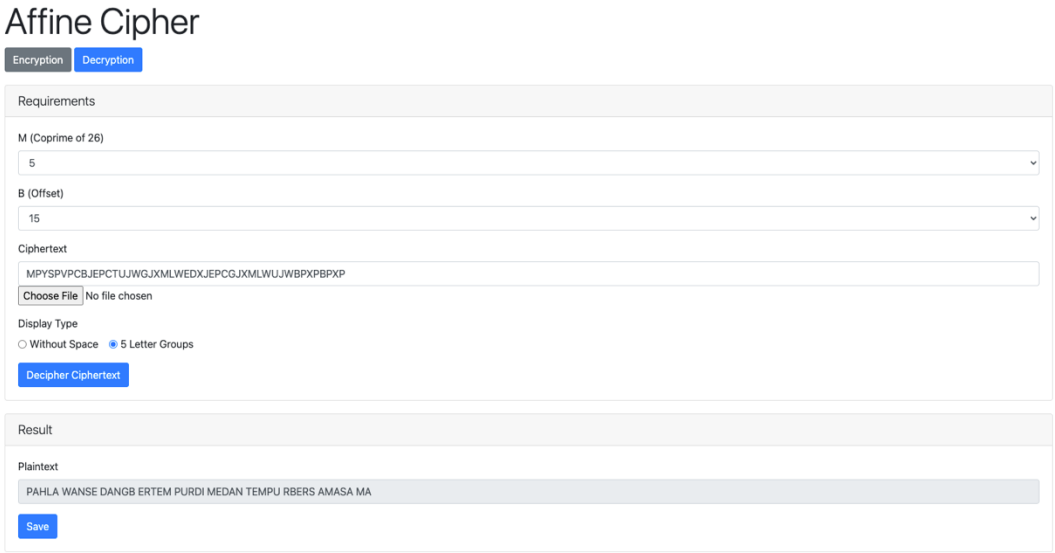
b. Dekripsi

M	5
B	15
Ciphertext	MPYSPVPCBJEPCTUJWGJXMLWEDXJEPCGJXMLWUJWBPXPBPXP

Hasil Tanpa Spasi



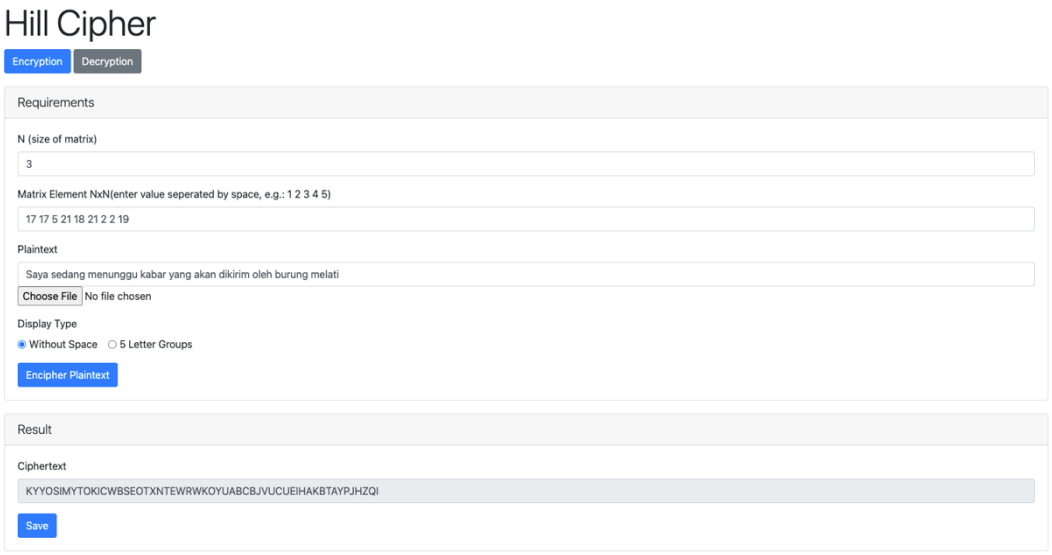
Hasil dalam Kelompok 5 Huruf



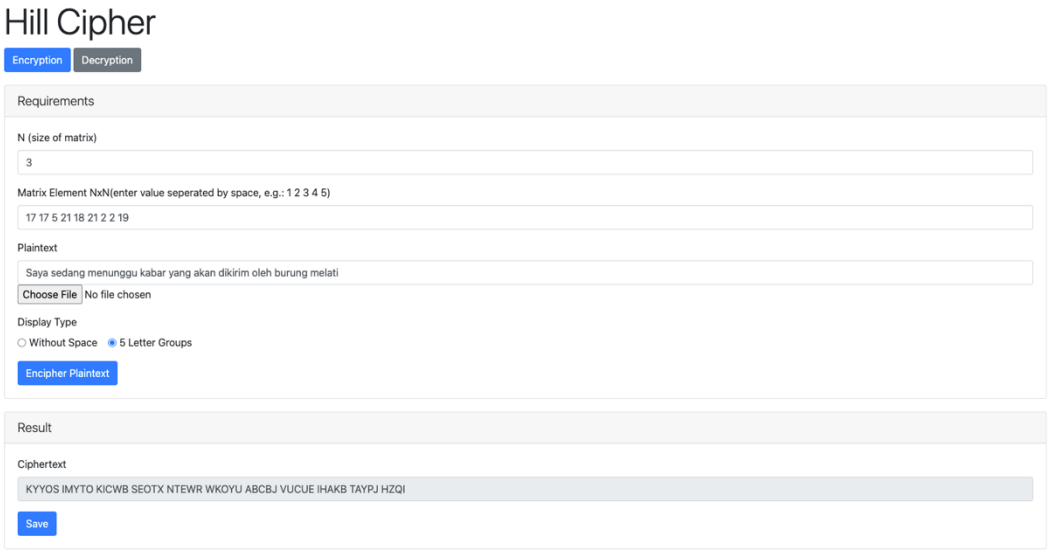
7. Hill Cipher

a. Enkripsi

N	3
Elemen Matriks	17 17 5 21 18 21 2 2 19
Plaintext	Saya sedang menunggu kabar yang akan dikirim oleh burung melati
Hasil Tanpa Spasi	



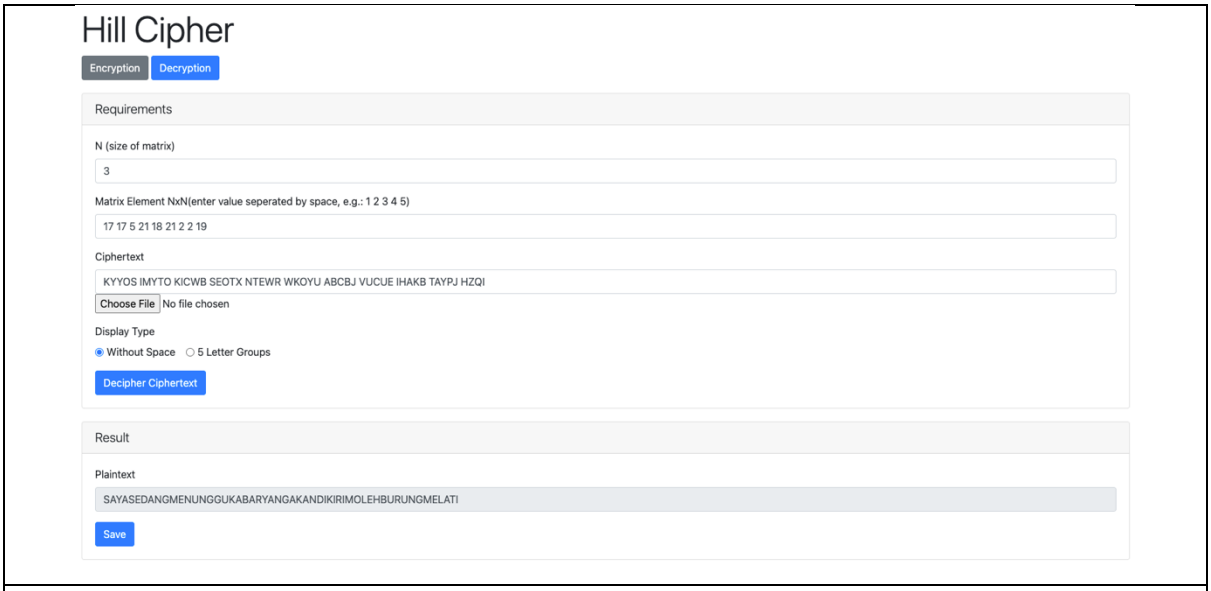
Hasil dalam Kelompok 5 Huruf



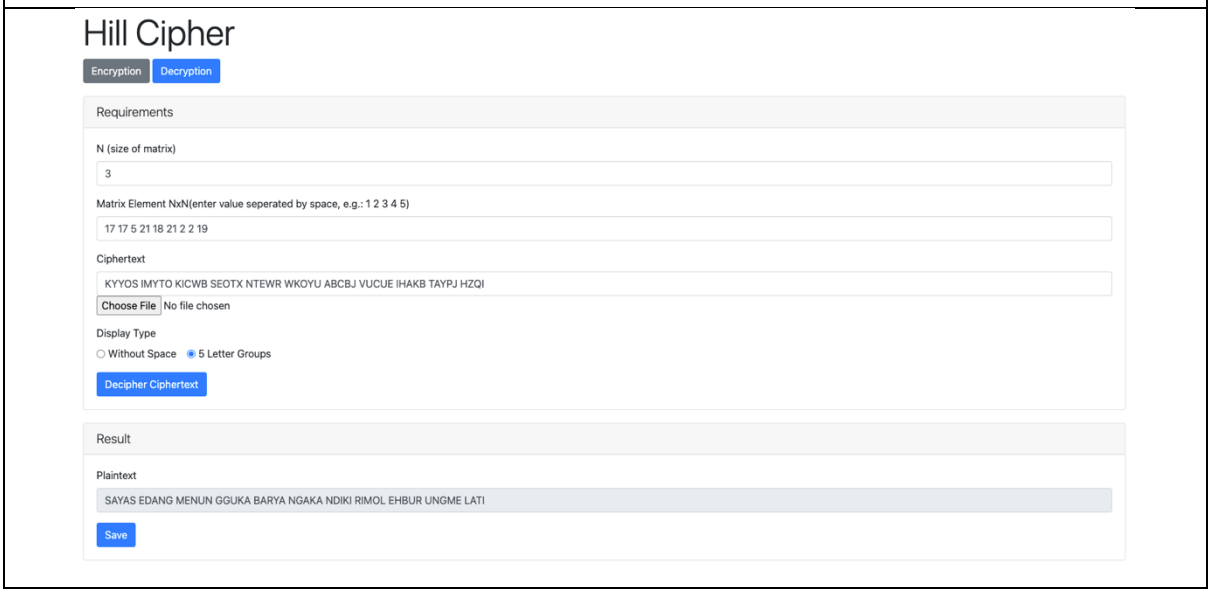
b. Dekripsi

N	3
Elemen Matriks	17 17 5 21 18 21 2 2 19
Ciphertext	KYYOS IMYTO KICWB SEOTX NTEWR WKOYU ABCBJ VUCUE IHAKB TAYPJ HZQI

Hasil Tanpa Spasi



Hasil dalam Kelompok 5 Huruf



BAB IV
Lampiran

1. **Tautan github yang berisi kode program**

   https://github.com/reymyr/Tucil-1-Kriptografi

2. **Cek list tugas**

No.	Spek	Berhasil (✓)	Kurang Berhasil(✓)	Keterangan
1.	Vigenere standard	✓		
2.	Full Vigenere Cipher	✓		
3.	Auto Key Vigenere Cipher	✓		
4.	Extended Vigenere Cipher	✓		
5.	Playfair Cipher	✓		Tidak bisa mendekripsikan ciphertext yang awalnya berisi huruf j (diganti dengan huruf I saat dienkripsi). Misalkan “JALAN” saat akan didekripsi Kembali menjadi “IALAN”
6.	Affine Cipher	✓		Masukan dibatasi oleh select option untuk M dan B, sehingga tidak diperlukan validasi di backend
7.	Hill Cipher	✓		Masukan matriks dan ukurannya harus sesuai serta harus bisa diinvers modulo. Karena baik

				backend maupun frontend tidak menangani kasus tersebut